REMARKS

Applicant is in receipt of the Office Action mailed March 30, 2010. Claims 1, 9, 11, 18, 19, 21-39, 43, 44, and 46-82 have been amended. Claims 1-82 are pending in the case. Reconsideration of the present case is earnestly requested in light of the following remarks.

Information Disclosure Statement:

Applicant notes that information disclosure statements with accompanying Forms PTO-1449 were submitted on February 7, 2003, May 16, 2003, and February 18, 2004. Applicant respectfully requests signed and fully initialed copies of the Forms PTO-1449 previously submitted on February 7, 2003, on May 16, 2003, and on February 18, 2004.

Section 102 Rejections

Claims 1-8, 11-18,21-26,29-34, and 37-38, and 39-82 are rejected under 35 U.S.C. 102(e) as being anticipated by Fowlow et al. (5,991,535, "Fowlow"). Applicant respectfully traverses the rejection.

Applicant respectfully reminds the Examiner that per MPEP 2143.03, all words in a claim must be considered in judging the patentability of that claim against the prior art." In re Wilson, 424 F.2d 1382, 1385, 165 USPQ 494, 496 (CCPA 1970). Additionally, the standard for "anticipation" is one of strict identity. Anticipation requires the presence in a single prior art reference disclosure of each and every element of the claimed invention, arranged as in the claim. M.P.E.P 2131; *Lindemann Maschinenfabrik GmbH v. American Hoist & Derrick Co.*, 221 USPQ 481, 485 (Fed. Cir. 1984). The **identical invention** must be shown in as complete detail as is contained in the claims. *Richardson v. Suzuki Motor Co.*, 9 USPQ2d 1913, 1920 (Fed. Cir. 1989).

Amended claim 1 recites:

1. A computer-implemented method for creating a graphical data flow program, wherein the graphical data flow program is configured to invoke a method of an object,

wherein the method for creating the graphical data flow program operates in a computer including a display and a user input device, the method for creating the graphical data flow program comprising:

displaying on the screen a node in the graphical data flow program in response to user input, wherein the node is configured to invoke a method of an object, wherein the graphical data flow program comprises a plurality of interconnected nodes which visually indicate functionality of the graphical data flow program, wherein the plurality of interconnected nodes are connected by lines which represent flow of data among the nodes;

configuring the node to receive information on the object in response to user input, wherein said configuring comprises connecting the information on the object to an input of the node;

wherein the node is configured to invoke the method of the object during execution of the graphical data flow program.

Nowhere does the cited art teach or suggest **displaying on the screen a node in the graphical data flow program in response to user input, wherein the node is configured to invoke a method of an object**, as recited in claim 1.

Regarding the specific feature "wherein the node is configured to invoke a method of an object". The Examiner asserts that "a node is merely a junction of some kind". This is incorrect. The node of claim 1 is an actual *graphical data flow program* element that is executable to invoke a method. Moreover, as explained in the previous Response, the cited "making of a selection action on one of the icons" refers to the user selecting an icon for inclusion on the worksheet, and has nothing to do with a node in a graphical data flow program being configured to invoke a method of an object. Nowhere does Fowlow teach or suggest a node as recited in claim 1.

The Office Action cites Fowlow's Title and Abstract, asserting that these portions of Fowlow disclose this feature. These citations read thusly:

Title:

> Visual composition tool for constructing application programs using distributed objects on a distributed object network

Abstract:

A method, apparatus, and program code visually constructs object-oriented application software to be installed on a distributed object system. The method of the invention includes the following steps. Initially, the method provides a catalog facility which contains components having references to pre-existing objects within a distributed object system. A component is selected from the catalog facility for inclusion in the application software. A part corresponding to the object referenced by the selected component is derived from the selected component. The part is then made available to an application construction environment. In this environment, the part can be linked to at least one other part that also references a pre-existing object in the distributed object system. Graphical facilities are provided within the application construction environment for selecting and defining links among parts. Links define relationships between parts and thereby provide computer code for the application software under construction. When the part is linked to another part in the application construction environment, computer code is generated that will be effective to implement the relationship between the parts when the application program is run.

Applicant respectfully notes that these citations nowhere mention or even hint at a node in a program that can be configured to invoke a method of an object. Applicant submits that Fowlow describes the cited "parts" (that correspond to objects) being linked to define "relationships between parts and thereby provide computer code for the application software under construction", where "When the part is linked to another part in the application construction environment, computer code is generated that will be effective to implement the relationship between the parts when the application program is run." However, nowhere does this text, nor Fowlow in general, discuss or suggest a node in a program that is configured to invoke a method of an object.

Thus, Fowlow fails to disclose these features of claim 1.

Additionally, as explained previously, the Fowlow reference relates to a method for visually constructing object oriented text-based application software that is to be installed on a distributed object system. Applicant respectfully notes that Fowlow's worksheet and connected icons implement an editing/design mechanism whereby the user can specify or design a program, but which itself does not constitute a program.

For example, Figure 5 of the Fowlow patent illustrates a composition design environment which is evidently used for visually constructing object oriented (text-based) application software. As shown in Figure 5 of Fowlow and discussed beginning at column 11 line 1:

> "The connections between parts and other parts or interfaces is made using plugs such as shown at 544, and sockets such as shown at 546. … a socket is a representation of a service provided by an object, comprising usually an object reference that is passed by that object to another requesting object. A plug, conversely, is a service that an object is capable of requesting and processing. As will be known to those with skill in the object programming arts, objects communicate amongst themselves by passing and operating upon object references which communication is represented schematically by drawing connections (such as connection 536) between the plug of a first object and a socket of a second object."

Per this text, the connections between object icons described in the Fowlow reference represent object references that are used to request services as represented by a "socket". Nowhere are the icons (including sockets and plugs) and lines described as source code for a program; rather, these elements specify desired functionality and relationships of predefined program objects, where the icons correspond to or represent predefined textual code. Note that in Fowlow's approach, once the user has selected and linked the icons in the worksheet, a code generator *generates source code* (source files) for the (text-based) distributed object-oriented application program. Thus, Fowlow's worksheet/diagram is used to generate source code, but is not (textual or graphical) source code.

Applicant respectfully submits that the Fowlow reference does not teach the notion of a graphical dataflow program "wherein the graphical data flow program comprises a plurality of interconnected nodes which visually indicate functionality of the graphical data flow program, wherein the plurality of interconnected nodes are connected by lines which represent flow of data among the nodes", as recited in the present claims. A graphical data flow program comprises a ***program*** that includes a plurality of nodes or icons, wherein the nodes or icons are interconnected by lines, and wherein the lines represent the flow of data between the nodes. As those of skill in the programming arts know, a program generally includes some kind of source code, which is compiled or interpreted to produce an executable which may then be executed by a processor.

The nature of Fowlow's worksheet and connected icons is made clear throughout the Specification and Figures, where Fowlow's worksheet is disclosed as a form of editor for specifying application functionality, where the worksheet, once configured (with connected icons representing program components), is used to *generate source code for the application*. In other words, Fowlow's worksheet/diagram is a means for specifying a program, but is not itself a program, as the term is commonly understood by those of skill in the art.

More specifically, Fowlow discloses a "composition builder and associated interface", which includes the worksheet, which is used to specify an application, but does not produce or implement program source code. For example, cited Figure 4 clearly shows Fowlow's "composition builder" 402 providing its output to a code generator 408, which constructs program source files 410, i.e., source code, based on predefined text-based program templates 406. Applicant respectfully submits that these generated program source files 410 compose the program's source code (and thus may be considered the program), which is then compiled and linked by the ODF compiler/linker 414 to generate object software 416 and network applications 418, which may be considered to be the executable form of the program (application). This aspect of Fowlow is further emphasized in Figure 9 and related text, which illustrates and explains that once the user has specified the composition, source code for the program is generated and compiled, as shown in element 918. Nowhere does Fowlow describe the worksheet with connected icons as a compilable/executable program.

The Office Action asserts that "dataflow is merely the flow of data throughout a system", arguing that Fowlow teaches this feature, citing Figure 4 (discussed above), col.2:47-52, and col.3:30-59. The term "data flow" (or "dataflow") is a term of art in the programming domain, and does not simply mean "the flow of data" in the generic sense.

Rather, as is known by those of skill in the programming arts, a data flow (or dataflow) program is a program that executes according to the data flow programming paradigm, where each function executes whenever its required inputs are available. A *graphical data flow program* is a graphical program that executes according to the data flow programming paradigm. Fowlow has nothing whatsoever to do with data flow programs or data flow programming. Again, the nodes in a graphical data flow are actual

program elements, i.e., graphical source code, *not* simply icons that are used to generate text-based source code for a program.

Cited col.2:47-52 reads:

> Thus, in the development of large object-oriented applications the programmer must become aware of the interrelationships among the objects in the program, which awareness is made more difficult by the large amounts of textual material that must be absorbed and analyzed.

As may be seen, this text makes no mention or hint of a graphical data flow program. Applicant again respectfully notes that a worksheet or diagram illustrating or specifying the "flow of data" (Examiner's words) is not equivalent to a graphical data flow *program*.

Cited col.3:30-59 reads:

> The part is then made available to an application construction environment in which the part can be linked to at least one other part referencing a pre-exising object in the distributed object system using facilities for selecting and defining links among the parts to define relationships thereamong to provide thereby computer code for the object-oriented application software. Finally, the part is linked to at least one other part in the application construction environment to define thereby a relationship among the parts such that computer code effective to implement the relationship is generated when the application program is run.
>
> In one embodiment, the references to the pre-existing objects comprise icons, the catalog and application construction environment comprise graphical user interfaces, and the step of selecting includes making a selection action on one of the icons. In another embodiment, the step of making a reference to the pre-existing object available to the application construction environment comprises dragging the reference to from the catalog facility to the application construction facility. In still another embodiment, each of the parts each comprises plugs and sockets and the step of linking comprises defining a connection between a plug on a first part and a socket on a second part. In yet another embodiment, the catalog facility includes regions for viewing the icons and information regarding the pre-existing objects represented by the icons, and the step of selecting a component is effective to cause the display of information relating to the pre-existing object referred to by the selected component.

Cited col. 11 lines 1-22 reads:

> The connections between parts and other parts or interfaces is made using plugs such as shown at 544, and sockets such as shown at 546. As discussed in co-pending U.S. patent application Ser. No. 08/675,094 filed on even date herewith and incorporated herein by reference for all purposes, a socket is a representation of a service provided by an object, comprising usually an object reference that is passed by that object to another requesting object. A plug, conversely, is a service that an object is capable of requesting and processing. As will be known to those of skill in the object programming arts, objects communicate amongst themselves by passing and operating upon object references which communication is represented schematically by drawing connections (such as connection 536) between the plug of a first object and a socket of a second object. As will be apparent from the interface illustrated in FIG. 5, the present invention facilitates the construction of applications by taking advantage of the above-described programming object connection paradigm by providing a graphical environment in which icons representing objects (i.e., parts) are connected by using interactive tools to define connections between plugs and sockets.

As the above text makes clear, the icons, sockets, and plugs and their connections are a *specification* or *representation* from which textual source code that implements the described passing of object references between program objects is generated, but are not themselves source code, and thus, do not compose a program, much less, a graphical data flow program, as claimed.

Col.5:?(numeral missing in Office Action citation)-col.6:4 appears to broadly describe computer based implementations or media covering the primary categories of claims (system, (computer based)method, and memory medium) reciting Fowlow's invention, but does not appear to be germane to the actual functionality described elsewhere in Fowlow. The Examiner's assertion that "designed or constructed program instructions inherently provides for data flow from one instruction to the next to enable the program to execute properly and to completion to create new implementations for objects" (per col. 6:32-37) is incorrect at least because such standard information transfer or access between program instructions does not properly qualify as "data flow", per the accepted technical meaning of the term in the programming arts. In particular, Fowlow

does not teach a graphical program with lines that represent flow of data among the nodes.

Claims 26 and 16 of Fowlow recite graphical means for creating object-oriented application software, but clearly indicate that the graphically specified relationships and objects are not implemented until program source code is generated from the worksheet. Nowhere does Fowlow indicate that the worksheet (diagram with connected icons) itself is an executable program or source code of a program.

Thus, Fowlow fails to disclose a graphical program, nor a graphical program wherein lines interconnecting nodes in the graphical program specify data flow among the nodes, and, more particularly, fails to disclose a graphical data flow program with a node as claimed.

Nor does Fowlow disclose **configuring the node to receive information on the object in response to user input, wherein said configuring comprises connecting the information on the object to an input of the node**, as recited in claim 1.

Cited col.4:50-53 and adjacent text reads:

> Also included are program code devices for selecting a reference for inclusion in the application software from the catalog facility and making the reference to the pre-existing object available to the application construction environment. Furthermore, the computer program product includes additional code devices for linking the reference to at least one other reference in the application construction environment to define thereby a relationship between the references such that computer code effective to implement the relationship is generated when the application program is run.

First, as explained above, Fowlow's icons in no way teach the node of claim 1. Nor is Fowlow's worksheet (graphical diagram, e.g., per Figure 5) equivalent to a graphical program, nor, more specifically, a graphical data flow program. The citation is unclear as to the nature of the "application software", the "computer code", the "application program", and the relationships between them. For example, the text (and elsewhere in Fowlow, e.g., the Abstract) seems to indicate that once object references (represented by icons) are included in the application software and linked to define relationships between them, the "application program" is run, thereby generating

computer code implementing the specified relationships. However, the computer code is clearly described as implementing the functionality and relationships specified, and as being generated by the "code generator" based on the graphically specified relationships (worksheet) using program templates (see, e.g., Figures 4, 8, and 9, and related text, among others).

Thus, since the computer code, i.e., source code of the program, is generated (by the code generator) based on the graphical construction on the worksheet (diagram with icons and connections), the worksheet/diagram cannot be the program. Nor are the cited icons equivalent to the node of claim 1, as explained above. Thus, Fowlow does not, and cannot, disclose *configuring the node to receive information on the object in response to user input, wherein said configuring comprises connecting the information on the object to an input of the node.*

Nor does Fowlow disclose **wherein the node is configured to invoke the method of the object during execution of the graphical data flow program**, as recited in claim 1.

Cited col.3:35-41, and col.4:51-56, both quoted and discussed above, never mention or even hint at a node in a graphical data flow program that is configured to invoke a method of an object during execution of the data flow program. Rather, these citations, and Fowlow in general, disclose a worksheet (diagram) with linked icons that specify or represent desired relationships among references to text-based (object-oriented) program objects, where a code generator generates program source code (again, textual) implementing the specified relationships. As discussed above at length, Fowlow's diagram is not a program at all, nor, more specifically a graphical data flow program. Nor are any of Fowlow's icons capable of being configured to invoke a method of an object during execution of a graphical data flow program.

Thus, Fowlow also fails to teach or suggest these features of claim 1.

Thus, for at least the above reasons, Applicant submits that that the cited art fails to teach or suggest all the features and limitations of claim 1, and so claim 1, and those

claims respectively dependent therefrom, are patentably distinct and non-obvious over the cited art, and are thus allowable.

Independent claim 21 includes similar limitations as claim 1, and so the above arguments apply with equal force to this claim. Thus, for at least the above reasons, Applicant submits that claim 21, and those claims respectively dependent therefrom, are patentably distinct and non-obvious over the cited art, and are thus allowable.

Independent claims 11 and 29 also includes many of the same limitations as claim 1, but recites that the graphical data flow program and the node are each configured to invoke a property of an object, and that once configured, the node is configured to invoke the property of the object during execution of the graphical data flow program. Per the arguments presented above with respect to claim 1, Fowlow fails to disclose a graphical data flow program, nor a node in such a program, at all, and so Fowlow does not, and cannot, disclose a node in a graphical data flow program that invokes a property of an object during execution of the graphical data flow program.

Independent claims 37, 38, 39, 46, and 69 each recites a graphical data flow program that includes a node that either invokes a method of an object (claims 37, 39, 46, and 69) or invokes a property of an object (claim 38), during execution of the graphical data flow program, and so Fowlow also fails to teach all the features of these claims. Thus, for at least these reasons, claims 37, 38, 39, and 46, and those claims respectively dependent therefrom, are patentably distinct and non-obvious over the cited art, and are thus allowable.

The Fowlow reference also does not teach the notion of a node in a graphical dataflow program that is configurable to get or set a property of an object during execution of the graphical data flow program, as recited in claims 54, 61, and 76, and so these claims, and those claims respectively dependent therefrom, are also patentably distinct and non-obvious over the cited art, and are thus allowable.

Claims 9-10, 19-20, 27-28 and 35-36 were rejected under §103(a) as being unpatentable over Fowlow as applied to claim 1 above and further in view of Meyer. Applicant respectfully submits that these claims, whose respective independent claims were shown above to be allowable, are similarly allowable in view of the arguments

made with respect to Fowlow above. Further, Applicant notes that Meyer does not provide at least several of the missing elements from the Fowlow reference. For example, the Meyer patent is directed toward developing "a graphical control flow structure". The Meyer patent is based on the well known Grafcet/IEC 1131 standard, which is a standard for creating control flow diagrams. The Grafcet/IEC 1131 standard is not related to data flow graphical programs. Thus the Meyer reference does not teach or suggest the concept of a graphical data flow program as recited in the present claims.

Applicant also asserts that numerous other ones of the dependent claims recite further distinctions over the cited art. However, since the independent claims have been shown to be patentably distinct, a further discussion of the dependent claims is not necessary at this time.

Removal of the section 103 Rejection of the claims is earnestly requested.

CONCLUSION

In light of the foregoing amendments and remarks, Applicant submits the application is now in condition for allowance, and an early notice to that effect is requested.

If any extensions of time (under 37 C.F.R. § 1.136) are necessary to prevent the above-referenced application(s) from becoming abandoned, Applicant(s) hereby petition for such extensions. The Commissioner is hereby authorized to charge any fees which may be required or credit any overpayment to Meyertons, Hood, Kivlin, Kowert & Goetzel P.C., Deposit Account No. 50-1505/5150-18302/JCH.

Also filed herewith are the following items:

☒ Request for Continued Examination

☐ Terminal Disclaimer

☐ Power of Attorney By Assignee and Revocation of Previous Powers

☐ Notice of Change of Address

☐ Other:

Respectfully submitted,

/Jeffrey C. Hood/

Jeffrey C. Hood, Reg. #35198
ATTORNEY FOR APPLICANT(S)

Meyertons, Hood, Kivlin, Kowert & Goetzel PC
P.O. Box 398
Austin, TX 78767-0398
Phone: (512) 853-8800
Date: 2010-06-28 JCH/MSW